Lesson 6

Misc & Finale

Haoxiang Sun @ RUC CS

re

- 正则表达式库（自带）
- https://regexr.com/
- 例子由 o1-preview 生成

demo

```python
import re

text = """
您好，我的手机号码是138-0013-8000，请随时联系我。
另外，你也可以拨打办公室电话：010 8888 9999。
如果我不在，请联系我的同事，电话是(020)-1234-5678。
"""

# 定义正则表达式模式，匹配不同格式的电话号码
phone_pattern = re.compile(r"""
    (\+?\d{1,3}[\s-]?)?      # 可选的国家代码，如+86
    (\(?\d{2,4}\)?[\s-]?)    # 区号，可能带括号
    (\d{4}[\s-]?\d{4})       # 号码的后8位
""", re.VERBOSE)

# 查找所有匹配的电话号码
phones = phone_pattern.findall(text)

# 格式化并打印电话号码
formatted_numbers = []
for phone in phones:
    # 合并匹配的组，并去除多余的符号
    number = ''.join(phone).replace('(', '').replace(')', '').replace(' ',
'').replace('-', '')
    formatted_numbers.append(number)

print("提取的电话号码：")
for num in formatted_numbers:
    print(num)
```

datetime / time

- 有关时间的两个库
- 可以帮助你记录日志
- 例子由 o1-preview 生成

demo

```python
import datetime
import time
import os

def countdown_to_new_year():
    while True:
        # 获取当前时间
        now = datetime.datetime.now()
        # 定义下一年的元旦零点
        next_year = now.year + 1
        new_year = datetime.datetime(next_year, 1, 1, 0, 0, 0)
        # 计算剩余时间
        time_left = new_year - now
        # 如果时间到了，退出循环
        if time_left.total_seconds() <= 0:
            print("新年快乐！")
            break
        # 提取天、小时、分钟和秒
        days = time_left.days
        hours, remainder = divmod(time_left.seconds, 3600)
        minutes, seconds = divmod(remainder, 60)
        # 清屏（根据操作系统选择命令）
        os.system('cls' if os.name == 'nt' else 'clear')
        # 显示倒计时
        print(f"距离 {next_year} 年新年还有：{days} 天 {hours} 小时 {minutes} 分
{seconds} 秒")
        # 每秒更新一次
        time.sleep(1)

if __name__ == "__main__":
    countdown_to_new_year()
```

fractions

- 原生分数表示
- 例子由 o1-preview 生成

demo

```python
from fractions import Fraction

def egyptian_fraction(numerator, denominator):
    frac = Fraction(numerator, denominator)
    denominators = []

    while frac.numerator != 0:
        # 计算当前需要的最小单位分数
        unit_denominator = (frac.denominator + frac.numerator - 1) // frac.numerator
        denominators.append(unit_denominator)

        # 减去这个单位分数
        frac -= Fraction(1, unit_denominator)

    return denominators

# 示例使用
numerator = 4
denominator = 13
result = egyptian_fraction(numerator, denominator)

print(f"{numerator}/{denominator} 可以表示为以下单位分数之和: ")
print(" + ".join(f"1/{d}" for d in result))
```

pypandoc

- Pandoc 文档转换的 Python 接口
- 支持海量格式的相互转换
- 用途：用来读取 docx 文件或者其他文件

Logging

- 日志库

math / random / statistics

- 各种科学计算和统计功能

demo

```python
import random
import math
import statistics

def estimate_pi(num_points):
    inside_circle = 0
    for _ in range(num_points):
        x = random.uniform(-1, 1)  # 在-1到1之间生成随机浮点数
        y = random.uniform(-1, 1)
        distance = math.hypot(x, y)  # 计算点(x, y)到原点(0, 0)的距离
        if distance <= 1:
            inside_circle += 1
    return (inside_circle / num_points) * 4

# 设置模拟次数和每次模拟的点数
num_simulations = 1000
num_points_per_simulation = 1000

# 存储每次模拟的π值估计
pi_estimates = []

for _ in range(num_simulations):
    pi_estimate = estimate_pi(num_points_per_simulation)
    pi_estimates.append(pi_estimate)

# 使用statistics模块计算平均值和标准差
mean_pi = statistics.mean(pi_estimates)
stdev_pi = statistics.stdev(pi_estimates)

print(f"经过{num_simulations}次模拟，每次使用{num_points_per_simulation}个点，")
print(f"估计的π值平均值为：{mean_pi}")
print(f"估计的π值标准差为：{stdev_pi}")
```

Scipy

- 一个用于数学、科学、工程领域的常用软件包，可以处理最优化、线性代数、积分、插值、拟合、特殊函数、快速傅里叶变换、信号处理、图像处理、常微分方程求解器等
- 可以进行假设检验，etc.

demo

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# 定义正态分布的参数
mu = 0    # 均值
sigma = 1 # 标准差

# 生成 x 轴的数据点
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)

# 计算概率密度函数和累积分布函数
pdf = norm.pdf(x, mu, sigma)
cdf = norm.cdf(x, mu, sigma)

# 绘制 PDF 和 CDF
plt.figure(figsize=(10,5))

plt.subplot(1, 2, 1)
plt.plot(x, pdf, label='PDF')
plt.title('正态分布概率密度函数')
plt.xlabel('x')
plt.ylabel('概率密度')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(x, cdf, color='red', label='CDF')
plt.title('正态分布累积分布函数')
plt.xlabel('x')
plt.ylabel('累计概率')
plt.legend()

plt.tight_layout()
plt.show()
```

argparse

- 用来处理命令行参数

demo

```python
import argparse

def main():
    parser = argparse.ArgumentParser(description='简单命令行计算器')
    parser.add_argument('num1', type=float, help='第一个数字')
    parser.add_argument('operator', type=str, choices=['+', '-', '*', '/'], help='运算符（+、-、*、/）')
    parser.add_argument('num2', type=float, help='第二个数字')
    parser.add_argument('-v', '--verbose', action='store_true', help='输出详细计算过程')

    args = parser.parse_args()

    if args.operator == '+':
        result = args.num1 + args.num2
        op = '+'
    elif args.operator == '-':
        result = args.num1 - args.num2
        op = '-'
    elif args.operator == '*':
        result = args.num1 * args.num2
        op = '*'
    elif args.operator == '/':
        result = args.num1 / args.num2
        op = '/'

    if args.verbose:
        print(f'{args.num1} {op} {args.num2} = {result}')
    else:
        print(result)

if __name__ == '__main__':
    main()
```

Pandas

- 通常围绕 dataframe 来展开

Pandas 是一个开源的数据分析和数据处理库，它是基于 Python 编程语言的。
Pandas 提供了易于使用的数据结构和数据分析工具，特别适用于处理结构化数据，如表格型数据（类似于Excel表格）。
Pandas 是数据科学和分析领域中常用的工具之一，它使得用户能够轻松地从各种数据源中导入数据，并对数据进行高效的操作和分析。
Pandas 主要引入了两种新的数据结构：**DataFrame** 和 **Series**。

demo

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1. 创建数据

# 创建产品列表
products = ['手机', '笔记本电脑', '平板电脑', '耳机', '智能手表']

# 创建数据字典
data = {
    '订单ID': range(1001, 1011),
    '产品': np.random.choice(products, 10),
    '数量': np.random.randint(1, 5, size=10),
    '单价': np.random.uniform(100, 2000, size=10).round(2),
    '订单日期': pd.date_range(start='2023-01-01', periods=10, freq='D')
}

# 创建 DataFrame
df = pd.DataFrame(data)

# 计算总价
df['总价'] = df['数量'] * df['单价']

print("初始数据: ")
print(df)
```

```python
# 2. 数据保存与读取

# 保存为 CSV
df.to_csv('sales_data.csv', index=False)

# 从 CSV 读取，并解析订单日期为日期时间类型
df = pd.read_csv('sales_data.csv', parse_dates=['订单日期'])

print("\n从 CSV 读取的数据：")
print(df)

# 3. 数据探索与分析

# 查看前5行
print("\n数据的前5行：")
print(df.head())

# 数据信息
print("\n数据信息：")
print(df.info())

# 描述统计
print("\n描述统计：")
print(df.describe())

# 选择特定列
product_sales = df[['产品', '数量', '总价']]
print("\n产品销售数据：")
print(product_sales)
```

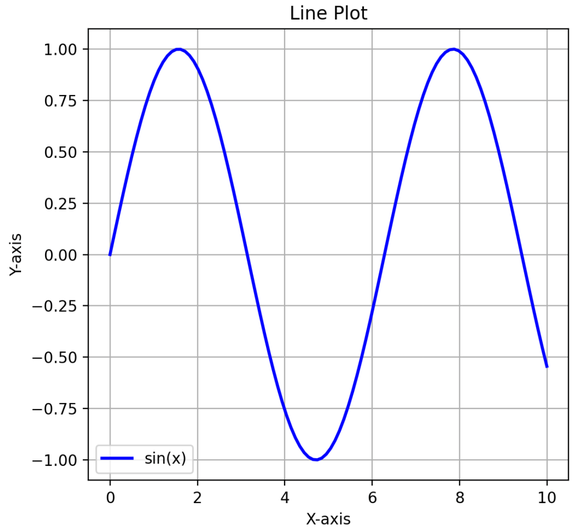
```python
# 过滤出销售数量大于等于3的订单
filtered_df = df[df['数量'] >= 3]
print("\n数量大于等于3的订单: ")
print(filtered_df)

# 处理缺失值

# 随机引入缺失值
df.loc[np.random.choice(df.index, 2), '单价'] = np.nan
print("\n引入缺失值后的数据: ")
print(df)

# 检测缺失值
print("\n缺失值统计: ")
print(df.isnull().sum())

# 填充缺失值
df['单价'].fillna(df['单价'].mean(), inplace=True)

# 重新计算总价
df['总价'] = df['数量'] * df['单价']
print("\n填充缺失值并重新计算总价后的数据: ")
print(df)

# 4. 数据分组与聚合

# 按产品分组，计算总销售额和销售数量
grouped = df.groupby('产品').agg({
    '数量': 'sum',
    '总价': 'sum'
}).reset_index()
```

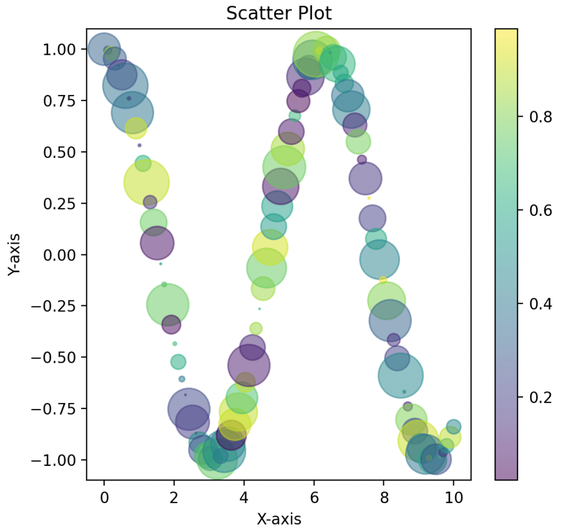
```python
print("\n按产品分组的聚合数据: ")
print(grouped)

# 5. 数据透视表

# 创建数据透视表
pivot_table = df.pivot_table(index='订单日期', columns='产品', values='数量', aggfunc='sum',
fill_value=0)
print("\n数据透视表: ")
print(pivot_table)

# 6. 数据合并

# 创建产品信息表
product_info = pd.DataFrame({
    '产品': products,
    '类别': ['电子产品', '电子产品', '电子产品', '配件', '配件'],
    '库存': [50, 30, 45, 100, 60]
})

print("\n产品信息表: ")
print(product_info)

# 合并数据
merged_df = pd.merge(df, product_info, on='产品', how='left')
print("\n合并后的数据: ")
print(merged_df)

# 7. 时间序列分析

# 设置订单日期为索引
df.set_index('订单日期', inplace=True)
```

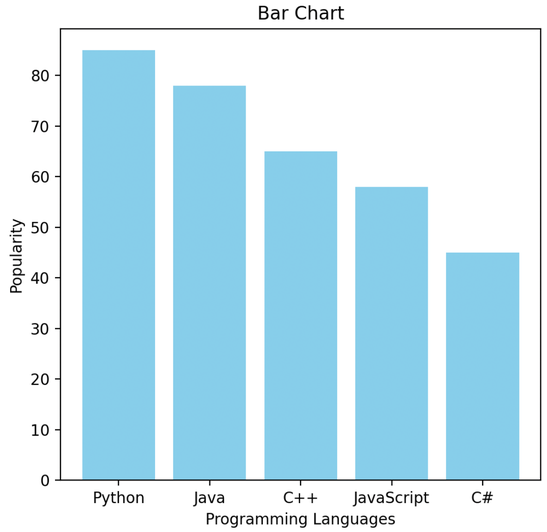
```python
# 按月统计销售总额
monthly_sales = df.resample('M')['总价'].sum()
print("\n月度销售总额: ")
print(monthly_sales)

# 8. 数据可视化
```

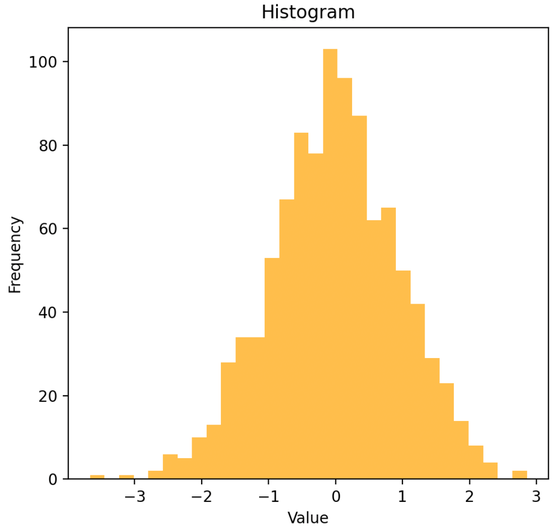
Matplotlib

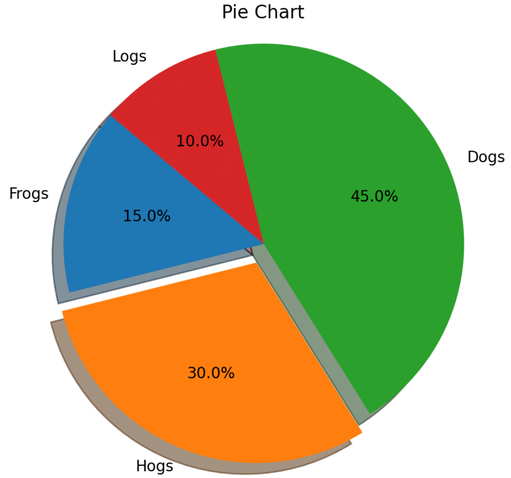
- 绘图

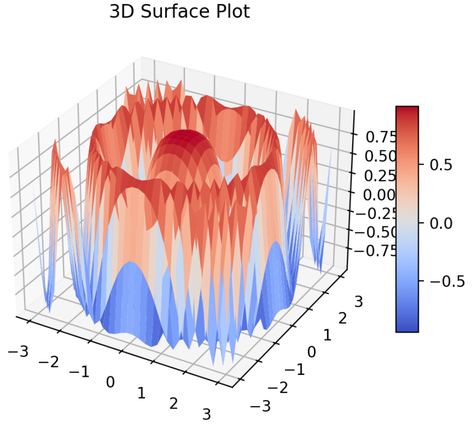












Beautiful Soup

- 用来 parse HTML / XML 文件

```
页面标题：新闻网站

文章 1:
标题：足球比赛精彩回顾
描述：昨晚的足球比赛真是扣人心弦。
链接：http://example.com/sports/football
类型：sports

文章 2:
标题：最新科技产品发布
描述：新一代智能手机即将上市。
链接：http://example.com/tech/newphone
类型：technology

文章 3:
标题：热门电影上映
描述：备受期待的大片终于上映了。
链接：http://example.com/entertainment/movie
类型：entertainment

体育类文章：
标题：足球比赛精彩回顾

修改后的第一篇文章标题：足球比赛最新报道

添加新文章后，所有文章标题：
标题：足球比赛最新报道
标题：最新科技产品发布
标题：热门电影上映
标题：今日天气预报

删除娱乐类文章后，所有文章标题：
标题：足球比赛最新报道
标题：最新科技产品发布
标题：今日天气预报

修改后的 HTML 文档：
<html>
 <head>
  <title>
   新闻网站
  </title>
 </head>
 <body>
  <h1>
   今日头条
  </h1>
  <div class="article" data-type="sports">
```

Requests

- 用来发送 http 请求

demo

```python
import requests

# 获取随机笑话的API地址
url = "https://official-joke-api.appspot.com/random_joke"

# 发送GET请求
response = requests.get(url)

# 将响应内容转换为JSON格式
joke = response.json()

# 打印笑话的 setup 和 punchline
print(f"笑话: {joke['setup']}")
print(f"答案: {joke['punchline']}")
```

Flask / Gradio

- 搭建属于自己的网站 / RESTful API 服务器

Pyvis

- 用来进行可视化

demo

```python
from pyvis.network import Network
import random

# 创建一个空的网络图对象
net = Network(notebook=False)

num_nodes = 50  # 节点数量
nodes = list(range(1, num_nodes + 1))

# 添加节点
for node in nodes:
    net.add_node(node, label=f'节点{node}')

# 随机添加边
num_edges = 100  # 边的数量，您可以根据需要调整
for _ in range(num_edges):
    n1, n2 = random.sample(nodes, 2)
    net.add_edge(n1, n2)

# 可选：添加物理引擎的设置按钮
# net.show_buttons(filter_=['physics'])

# 生成并保存HTML文件
net.show('random_graph.html', notebook=False)
```

Pillow

- 用来处理图像

demo

```python
from PIL import Image

# 创建一个新的 RGB 图像，尺寸为 (256, 256)
width, height = 256, 256
image = Image.new('RGB', (width, height))

# 生成彩色渐变
for x in range(width):
    for y in range(height):
        red = x   # 红色分量，取决于 x 坐标
        green = y # 绿色分量，取决于 y 坐标
        blue = (x + y) // 2 # 蓝色分量，为 x 和 y 的平均值
        image.putpixel((x, y), (red, green, blue))

# 保存生成的图像
image.save('gradient.png')

print("已成功生成渐变图像 gradient.png")
```

Sympy

- 符号计算库

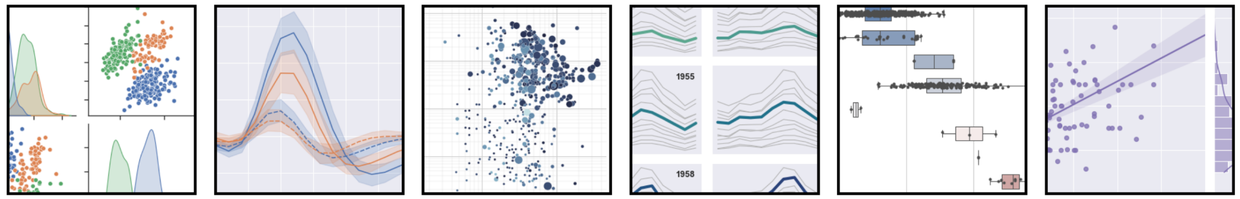
demo

```python
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt

# 定义符号变量
x = sp.symbols('x')
# 定义函数 f(x)
f = sp.ln(x) * sp.sin(x)
# 计算一阶导数 f'(x)
f_prime = sp.diff(f, x)
# 计算二阶导数 f''(x)
f_double_prime = sp.diff(f_prime, x)
# 显示计算结果
print("函数 f(x):")
sp.pprint(f)
print("\n一阶导数 f'(x):")
sp.pprint(f_prime)
print("\n二阶导数 f''(x):")
sp.pprint(f_double_prime)
# 将符号函数转换为数值计算的函数
f_lambdified = sp.lambdify(x, f, modules=['numpy'])
f_prime_lambdified = sp.lambdify(x, f_prime, modules=['numpy'])
f_double_prime_lambdified = sp.lambdify(x, f_double_prime, modules=['numpy'])
# 定义绘制范围，避免 x=0 以防 ln(0) 的数学错误
x_vals = np.linspace(0.1, 10, 400)
# 计算函数值
f_vals = f_lambdified(x_vals)
f_prime_vals = f_prime_lambdified(x_vals)
f_double_prime_vals = f_double_prime_lambdified(x_vals)
# 绘制函数和导数
plt.figure(figsize=(12,8))
plt.plot(x_vals, f_vals, label='$f(x) = \ln(x) \cdot \sin(x)$')
plt.plot(x_vals, f_prime_vals, label="$f'(x)$")
plt.plot(x_vals, f_double_prime_vals, label="$f''(x)$")
plt.title('函数及其导数')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

Seaborn

- 基于 matplotlib 的可视化库

seaborn: statistical data visualization



demo

```python
import seaborn as sns
import matplotlib.pyplot as plt

# 加载示例数据集
titanic = sns.load_dataset('titanic')

# 设置Seaborn的主题
sns.set_theme(style='whitegrid')

# 创建一个散点图并添加回归线
plt.figure(figsize=(12, 6))
scatter_plot = sns.scatterplot(
    data=titanic,
    x='age',
    y='fare',
    hue='survived',
    style='sex',
    palette='Set1',
    alpha=0.6,
    s=100
)
sns.regplot(
    data=titanic,
    x='age',
    y='fare',
    scatter=False,
    color='green',
    line_kws={'linewidth': 2}
)

# 设置图表标题和标签
plt.title('Titanic Passengers: Age vs Fare', fontsize=16)
plt.xlabel('Age', fontsize=14)
plt.ylabel('Fare ($)', fontsize=14)

# 设置图例
plt.legend(title='Survived & Sex', fontsize=12, title_fontsize=13, loc='upper right')

# 显示图表
plt.show()
```

课程内容回顾

- Lesson 1: 配置 miniconda 环境，基本语法，字符串，函数，lambda
- Lesson 2: 各种容器与数据结构，递归
- Lesson 3: 面向对象编程，大语言模型的加载与微调
- Lesson 4: CVXPY, Gurobi, scikit-learn 等优化包
- Lesson 5: AI 辅助编程介绍，numpy，playwrights
- Lesson 6: 杂项（各种常用的包的介绍）

Thank you for listening!

- hxiang.sun@gmail.com
- hxsun@ruc.edu.cn
- https://dl.coderbak.com/24fa_python/
- https://github.com/CoderBak
- Wechat: CoderBak